

Autotuning for Exascale Data Management

Kesheng Wu*, Ke-Jou Hsu, and Surendra Byna
Lawrence Berkeley National Laboratory

This position paper makes a case for autotuning in exascale data management systems. The strategy of autotuning has been demonstrated to be very effective for a wide variety of systems including math libraries [1][2][3], compilers [4], and commercial database systems [5][6]. We propose to utilize the same strategy in a new data management system that can dynamically adapt to the evolving storage hardware and shifting workload.

Introduction

High Performance Computers (HPC) typically store data onto some large parallel file systems attached through high-speed networks. The file system was initially designed many decades ago for systems with a single CPU and a single disk. This design is expected to be a bottleneck in many exascale applications [7][8], especially those Big Data applications [9][10][11][12]. These challenges arise from complex storage hardware as well as varying access patterns. The storage systems often consist of thousands of hard drives and hundreds of dedicated storage managers connected through specialized computer network. It is necessary to have detailed knowledge about many pieces of these components in order to achieve a good performance. As in many other such cases, the relationship between performance and the configuration parameters is highly irregular and the most effective way to locate the optimal parameter choices is to actually measure the performance on a specific system. This approach of systematically measuring the performance under various configuration parameters to determine the best choices is known as autotuning. We believe this strategy to be appropriate for addressing the hardware complexity.

A file is basically a sequence of bytes. After a file is written in a particular way, it is only effective for certain read access patterns. Let X , Y and Z denote the three dimensions of a 3D array. If it is written with Z as the fastest varying dimension, then reading a section of the array with fixed X and Y would be reading a consecutive block of the data file, which is efficient, however reading a section of the array with fixed Y and Z would be accessing elements that are spread apart from each other, which is time consuming. The database management systems (DBMS) have to deal with similar variations in the data access patterns. The database community has developed an extensive set of self-managing techniques [5][6][15]. These techniques are alternative forms of autotuning.

Autotuning in Scientific Computing

In scientific computing, the term autotuning was initially used to describe the strategy of applying a systematic testing of the myriad options to determine the best choices on a specific computer. This was demonstrated to be effective for common mathematical operations such as matrix-matrix multiplications [13][14]. In these case, the algorithms are adjust to have different blocking factors in order to take full advantages of hardware details, such as cache sizes, levels, cache line sizes, bandwidth of the caches, and number of registers of various types. These configuration parameters affect the performance in very complex ways, there is no easy way to determine the algorithmic choices to achieve the optimal performance. An effective strategy was to actually measure the performance every combination of algorithmic choices and record the best options for different problem sizes. This strategy is generally known as autotuning, and has been successfully used in a large variety of applications [1][2][3][4]. The general strategy of letting users specify the “what” and have systems figure out the “how” has been applied to a wide variety of problems. DBMS is well-known example.

* Contact author. Email KWu@lbl.gov.

Self-Managing DBMS

DBMS generally requires the users to describe their data as relational tables and specify their analysis tasks in the Structure Query Language (SQL). The relational data model is a logical model of data and SQL essentially specifies what to do instead of how to complete a task. The DBMS software is free to decide the physical organization of user data and processes SQL statements in the most efficient manner it can find. Over the years, researchers have developed many data structures and data processing algorithms to accelerate the DBMS operations, while users continue to execute their old SQL programs.

We notice two important approaches from the successes of DBMS systems: a high-level data model and autotuning. The relational data model has served database systems well, however, many other choices are also used in other application areas. For example, many Internet companies use NoSQL systems that organize their data as key-value pairs. In scientific applications, many have chosen high-level data formats implementing variations of the array model. A good way for our new data management system to gain acceptance is to follow one of these data formats and adopt the array data model.

In database literature, the autotuning approach is generally referred to as self-tuning or self-management. There are implementations of this feature in commercial DBMS products for nearly 20 years [5][15]. In this regard, autotuning is a proven technology for data management. Our key challenge is find an effective implementation strategy.

Early Experiences in Autotuning for Scientific Data Management

To minimize the entry barrier, we want the new scientific data management system to have a familiar user interface. Since the bulk of the scientific data is in one of the popular high-level data formats, a natural choice is to use the interface of one of these data format libraries. Among the data formats, HDF5 and netCDF are the two with the largest amount of data in use. In particular, the majority of the climate modeling community uses netCDF due to an international agreement through IPCC. However, the recent version of netCDF has switched to use HDF5 as the basis. Therefore, HDF5 is in a unique position with the largest user community. There are a number of other data formats in use, but none of them has more users than HDF5. For example, FITS is used by the astronomy community, and ADIOS BP is used by a number of large applications running on some of the fastest HPC systems.

There are a number of other recent developments that make HDF5 a suitable choice for the proposed scientific data management system. HDF5 is developing autotuning features under the ExaHDF5 project and HDF5 has implemented a Virtual Object Layer that will make it possible for us to replace the decisions on the physical organization of user data. The on-going autotuning work in HDF5 can address the automatic selection of parameters to maximize the write performance on a given hardware system. However, because the final product is still a file, it is not able to adapt to the varying data access patterns.

To adapt to varying access patterns, we need a persistent service that can observe user access patterns, analyze the variations in them, and then decide what could be done to accommodate the new access patterns. We are in the process of designing the plug-in to HDF5 that will provide such functionality.

In summary, we believe autotuning to be the essential feature for an efficient scientific data management system. A new system is necessary because the existing scientific data is stored in files that cannot evolve dynamically when the usage patterns change. The autotuning approach has been extensively used in a number of different systems, and is therefore a proven strategy. We have chosen to implement this strategy through HDF5 Virtual Object Layer. This implementation strategy allows us to make use of a widely used data access interface and provide a smooth transition pass for a wide community of scientific applications.

References

- [1] D H Bailey *et al.* 2008. PERI auto-tuning. *J. Phys.: Conf. Ser.* **125** 012089 DOI=[10.1088/1742-6596/125/1/012089](https://doi.org/10.1088/1742-6596/125/1/012089).
- [2] V. Volkov and J. Demmel. 2008. Benchmarking GPUs to tune dense linear algebra. In SC08. DOI=[10.1109/SC.2008.5214359](https://doi.org/10.1109/SC.2008.5214359).
- [3] C Chan, *et al.* 2009. Autotuning multigrid with PetaBricks. In SC09. DOI=[10.1145/1654059.1654065](https://doi.org/10.1145/1654059.1654065).
- [4] A Tiwari, *et al.* 2009. A scalable auto-tuning framework for compiler optimization. In IPDPS 2009. DOI= [10.1109/IPDPS.2009.5161054](https://doi.org/10.1109/IPDPS.2009.5161054).
- [5] S Chaudhuri (Editor): IEEE CS Data Engineering Bulletin, Special Issue on Self-Tuning Databases and Application Tuning, [Vol.22 No.2](#), June 1999.
- [6] S Babu and K-U Sattler (Editors): IEEE CS Data Engineering Bulletin, Special Issue on Self-Managing Databases: Lessons Learned and Challenges Ahead. [Vol.34 No.4](#), Dec 2011.
- [7] I Raicu, I T Foster, and P Beckman. 2011. Making a case for distributed file systems at Exascale. In LSAP '11. DOI=[10.1145/1996029.1996034](https://doi.org/10.1145/1996029.1996034).
- [8] G Aloisio and S Fiore. 2009. Towards Exascale Distributed Data Management. *International Journal of High Performance Computing Applications*. v23: 398-400. DOI=[10.1177/1094342009347702](https://doi.org/10.1177/1094342009347702).
- [9] A Jacobs. 2009. The pathologies of big data. *Commun. ACM* 52, 8 (August 2009), 36-44. DOI=[10.1145/1536616.1536632](https://doi.org/10.1145/1536616.1536632).
- [10] C Lynch. 2008. Big data: How do your data grow? *Nature* 455, 28-29 (4 September 2008). DOI=[10.1038/455028a](https://doi.org/10.1038/455028a).
- [11] S Madden. 2012. From Databases to Big Data. *Internet Computing, IEEE*. V16(3), pp 4-6. DOI= [10.1109/MIC.2012.50](https://doi.org/10.1109/MIC.2012.50).
- [12] D Fisher, R DeLine, M Czerwinski, and S Drucker. 2012. Interactions with big data analytics. *interactions* 19, 3 (May 2012), 50-59. DOI=[10.1145/2168931.2168943](https://doi.org/10.1145/2168931.2168943).
- [13] R. Clint Whaley, A Petitet, and J Dongarra. Automated empirical optimizations of software and the ATLAS project. *Parallel Computing*, 27(1):3-25, 2001. DOI= [10.1016/S0167-8191\(00\)00087-9](https://doi.org/10.1016/S0167-8191(00)00087-9).
- [14] R Vuduc, J W Demmel and K A Yelick. 2005. OSKI: A library of automatically tuned sparse matrix kernels. *J. Phys.: Conf. Ser.* **16** 521. DOI=[10.1088/1742-6596/16/1/071](https://doi.org/10.1088/1742-6596/16/1/071).
- [15] D Zilio, S Lightstone, K Lyons, and G Lohman. 2001. Self-managing technology in IBM DB2 universal database. In CIKM '01, 541-543. DOI=[10.1145/502585.502682](https://doi.org/10.1145/502585.502682).